**RUBIO GARCIA RODRIGO.**

**CARLOS ENRIQUE MORAN GARABITO.**

**CINEMATICA DE ROBOTS.**

**TAREA 7CINEMATICA INVERSA CON MATRIZ HOMOGENEA.**

**8/A MECATRONICA.**

**UPZMG.**

****



**ITÁLICO.**

**Indica nuevos términos, direcciones URL, direcciones de correo electrónico, nombres de rutas y directorios, nombres de archivos, y extensiones de archivo.**

**ANCHO CONSTANTE.**

**Se utiliza para las listas de programas, así como dentro de los párrafos para referirse al programa elemental, mentores como nombres de funciones o variables, espacios de nombres, tipos de datos, entorno variables, declaraciones y palabras clave. También se utiliza para comandos, línea de comandos utilidades, y paquetes de ROS, nodos, temas, etc.**

**ANCHO CONSTANTE NEGRITA.**

**Muestra comandos u otro texto que debe ser escrito literalmente por el usuario.**

**ANCHO CONSTANTE ITÁLICA.**

**Muestra que el texto debe ser reemplazado por valores proporcionados por el usuario o por valores minado por el contexto.**

**EL GRAFICO DE ROS.**

**Uno de los "problemas de desafío" originales que motivaron el diseño de ROS fue conocido como el problema "buscar un objeto". Imagina un espacio relativamente grande y como robot plex con varias cámaras y escáneres láser, un brazo manipulador y una rueda base. En el problema "buscar un objeto", la tarea del robot es navegar a la casa típica o entorno de oficina, busque el artículo solicitado y envíelo a la ubicación solicitada. Esta tarea, al igual que muchas tareas de robótica, llevó a varias observaciones acerca de muchas robótica. Aplicaciones de software, que se convirtieron en algunos de los objetivos de diseño de ROS:**

**La tarea de la aplicación se puede descomponer en muchos subsistemas independientes, como la navegación, la visión por ordenador, el agarre, etc.**

**• Estos subsistemas se pueden usar para otras tareas, como hacer patrullas de seguridad, limpieza, entrega de correo, etc.**

**• Con el hardware adecuado y las capas de abstracción de geometría, la gran mayoría de los software de aplicación puede ejecutarse en cualquier robot.**

**Estos objetivos se pueden ilustrar mediante la representación fundamental de un sistema ROS: su gráfico, un sistema ROS se compone de muchos programas diferentes que se ejecutan simultáneamente y comunicarse unos con otros pasando mensajes. Es conveniente usar un gráfico matemático para representar esta colección de programas y mensajes: Los programas son los nodos gráficos, y los programas que se comunican entre sí son conectado por los bordes, de las primeras implementaciones de la aplicación "buscar un ítem" usando ROS, los detalles de este gráfico no son particularmente importantes; Sólo se proporciona para ilustrar el concepto general de un sistema ROS como una colección de nodos que pasan mensajes a uno otro podemos representar cualquier sistema ROS, grande o pequeño, de esta manera. De hecho, esta es la representación es tan útil para el desarrollo de software que en realidad nos referimos a ROS programas como nodos, para ayudarnos a recordar que cada programa es solo una parte de un sistema mucho más grande.**

**ROSCORE.**

**Roscore es un servicio que proporciona información de conexión a los nodos para que puedan transmitir mensajes entre sí. Cada nodo se conecta a roscore en el inicio para registrarse, los detalles de los flujos de mensajes que publica y los flujos a los que desea suscribir. Cuando aparece un nuevo nodo, Roscore le proporciona la información que necesita, debe establecer una conexión directa de igual a igual con otros nodos de publicación y subgrupos, escribiendo a los mismos temas de mensaje. Cada sistema ROS necesita un roscore en ejecución, ya que sin ella, los nodos no pueden encontrar otros nodos.**

**PENDIENTE.**

**catkin comprende un conjunto de macros CMake y scripts de Python personalizados para proporcionar más, funcionalidad sobre el flujo de trabajo normal de CMake. CMake es un abierto comúnmente usado sistema de compilación de origen. Si vas a dominar las sutilezas de amento, realmente ayuda si sabes un poco sobre CMake. Sin embargo, para el usuario de catkin más casual, todos ustedes lo que realmente necesito saber es que hay dos archivos, CMakeLists.txt y package.xml, que necesita agregar alguna información específica para que las cosas funcionen correctamente.**

**A continuación, llama a las diversas herramientas de catkin para generar los directorios y archivos que está utilizando, voy a necesitar mientras escribes código para tus robots. Estas herramientas serán introducidas como nosotros necesitamos a lo largo del libro. Antes de llegar a todo esto, sin embargo, necesitamos presentarte a los espacios de trabajo.**

**ESPACIOS DE TRABAJO.**

**Antes de comenzar a escribir cualquier código ROS, debe configurar un espacio de trabajo para que este código vivir en. Un espacio de trabajo es simplemente un conjunto de directorios en el que un conjunto relacionado de código ROS vive. Puede tener múltiples espacios de trabajo de ROS, pero solo puede trabajar en uno de ellos en cualquier momento. La forma sencilla de pensar esto es que solo puede ver el código que vive en tu espacio de trabajo actual.**

**PAQUETES DE ROS.**

**El software de ROS se organiza en paquetes, cada uno de los cuales contiene una combinación de código, datos y documentación.2 El ecosistema de ROS incluye miles de paquetes disponibles públicamente en repositorios abiertos, y muchos miles de paquetes más acechan detrás de los firewalls organizacionales. Los paquetes se encuentran dentro de los espacios de trabajo, en el directorio src. Cada directorio de paquetes debe incluir un archivo CMakeLists.txt y un archivo package.xml que describa el contenido del paquete y la forma en que Catkin debe interactuar con él, crear un nuevo paquete es fácil.**

**ROSRUN**

**Dado que ROS tiene una gran comunidad distribuida, su software está organizado en paquetes que son desarrollados independientemente por miembros de la comunidad. El concepto de una ROS el paquete se describirá con mayor detalle en los capítulos siguientes, pero un paquete puede debe considerarse como una colección de recursos que se crean y distribuyen juntos, paquetes edades son solo ubicaciones en el sistema de archivos, y debido a que los nodos ROS son típicamente programas ejecutables que se pueden cortar, uno podría rodear manualmente el sistema de archivos para iniciar todos los ROS, los nodos pueden estar profundamente enterrados en grandes jerarquías de directorios. Para automatizar esta tarea, ROS, proporciona una utilidad de línea de comandos llamada rosrun.**

**NOMBRES, ESPACIOS DE NOMBRES Y REASIGNACIÓN**

**Los nombres son un concepto fundamental en ROS. Nodos, flujos de mensajes (a menudo llamados "top-ics ”), y todos los parámetros deben tener nombres únicos. Por ejemplo, el nodo de la cámara en unrobot podría ser nombrado cámara, y podría mostrar un mensaje llamado imagen y lea un parámetro llamado frame\_rate para saber qué tan rápido enviar imágenes. Hasta ahora tan bueno. Pero, ¿qué pasa cuando un robot tiene dos cámaras? No querríamos tener que escribir un programa separado para cada cámara, tampoco querríamos la salida de ambas cámaras que se intercalarán en el tema de la imagen, ya que eso requeriría toda los que escriben la imagen tienen una lógica que separa las secuencias de imágenes.Más generalmente, las colisiones de espacios de nombres son extremadamente comunes en sistemas robóticos que a menudo contienen subsistemas de hardware o software idénticos para simplificar su ingeniería Neering, como los brazos, cámaras o ruedas idénticos izquierdo y derecho. ROS proporciona dos Mecanismos para manejar estas situaciones: espacios de nombres y reasignación.**

**ROSLAUNCH.**

**Roslaunch es una herramienta de línea de comandos diseñada para automatizar el lanzamiento de colecciones de los nodos ROS. En la superficie, se parece mucho a Rosrun, que necesita un nombre de paquete y un nombre de archivo.**

**LA TECLA DE TABULACIÓN.**

**Las herramientas de línea de comandos de ROS tienen soporte para completar pestañas. Cuando se utiliza rosrun, para por ejemplo, presionar la tecla Tab en medio de escribir el nombre de un paquete se completará automáticamente, para ti o, si hay varias finalizaciones potenciales, presionando nuevamente Tab le presentará una lista de posibles terminaciones. Al igual que con muchos otros comandos de Linux, el uso de tabulación con ROS le ahorrará una cantidad masiva de escritura, y ayuda a evitar errores de ortografía al intentar escribir nombres largos de paquetes o mensajes.**

**TRANSFORMADAS DE COORDENADAS.**

**La tarea “buscar un objeto” descrita en “El gráfico ROS” en la página 9 incluye muchas, hay muchos problemas que abordar, que abarcan casi todos los aspectos de la robótica y la artificial. Inteligencia (esa es una de las razones por las que se convirtió en un gran desafío para impulsar la diseño de ROS). Un problema que puede no ser inmediatamente obvio, pero es de suma importancia, es la gestión de marcos de coordenadas. En serio, coordinar los marcos son un gran problema en robótica.**

**POSES, POSICIONES, Y ORIENTACIONES.**

**Su robot promedio de recuperación de artículos tendrá un grupo de subsistemas, como un dispositivo móvil base, un escáner láser conectado a la base para permitirle navegar por el mundo, un cámara (visual y / o profundidad) unida a otra parte de la base para encontrar elementos para ser y un brazo manipulador con una mano que hará el agarre real de aquellos artículos. Un robot realmente bueno para recuperar objetos podría tener muchas más características, pero estas ya hay mucho para hacer marcos de coordenadas una preocupación importante.**

**EMPECEMOS CON EL LÁSER EN LA BASE.**

**Para interpretar correctamente un escaneo de rango producido por el láser, necesitamos saber exactamente en qué punto de la base está conectado el láser. Esta montado en el frente de la base? ¿La parte de atrás? ¿Está mirando hacia atrás? ¿Está montado al revés?(que no es infrecuente)? Más en general, podríamos preguntar: ¿cuál es la posición ¿Y orientación del láser respecto a la base?**